

#3  
03.2002

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Hirokazu KAWAMOTO, et al.

Application No.: 09/995,724

Filed: November 29, 2001

Examiner: Unassigned

Group Art Unit: 2171

March 12, 2002

# APPARATUS AND METHOD FOR CONTROLLING USER INTERFACE

Commissioner for Patents  
Washington, D.C. 20231

RECEIVED

MAR 14 2002

Technology Center 2100



## SUBMISSION OF PRIORITY DOCUMENTS

Sir:

In support of Applicants' claim for priority under 35 U.S.C. § 119, enclosed are certified copies of the following Japanese applications:

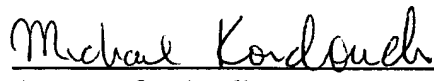
2000-364628, filed November 30, 2000;

2000-391377, filed December 22, 2000; and

2000-402713, filed December 28, 2000.

Applicants' undersigned attorney may be reached in our Washington, D.C. office by telephone at (202) 530-1010. All correspondence should continue to be directed to our below-listed address.

Respectfully submitted,

  
\_\_\_\_\_  
Attorney for Applicants  
Michael E. Kondoudis  
Registration No. 42,758

FITZPATRICK, CELLA, HARPER & SCINTO  
30 Rockefeller Plaza  
New York, New York 10112-3801  
Facsimile: (212) 218-2200

MEK/tmc

DC\_MAIN 89687 v 1

(translation of the front page of the priority document of  
Japanese Patent Application No. 2000-364628)

PATENT OFFICE  
JAPANESE GOVERNMENT

This is to certify that the annexed is a true copy of the  
following application as filed with this Office.

Date of Application: November 30, 2000

Application Number : Patent Application 2000-364628

Applicant(s) : Canon Kabushiki Kaisha

December 21, 2001

Commissioner,  
Patent Office

Kouzo OIKAWA

Certification Number 2001-3110586

CFM 1450 VS

日 本 国 特 許 庁

JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2000年11月30日

出 願 番 号

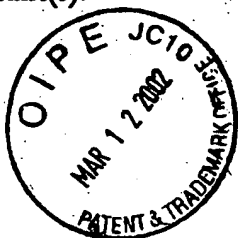
Application Number:

特願2000-364628

出 願 人

Applicant(s):

キヤノン株式会社



CERTIFIED COPY OF  
PRIORITY DOCUMENT

RECEIVED

MAR 14 2002

Technology Center 2100

2001年12月21日

特 許 庁 長 官  
Commissioner,  
Japan Patent Office

及 川 耕 造



【書類名】 特許願

【整理番号】 4338002

【提出日】 平成12年11月30日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 15/00

【発明の名称】 ユーザインタフェース制御装置および方法ならびに記憶媒体

【請求項の数】 23

【発明者】

    【住所又は居所】 東京都大田区下丸子3丁目30番2号 キヤノン株式会社  
社内

    【氏名】 有富 雅規

【特許出願人】

    【識別番号】 000001007

    【氏名又は名称】 キヤノン株式会社

【代理人】

    【識別番号】 100076428

    【弁理士】

    【氏名又は名称】 大塚 康德

    【電話番号】 03-5276-3241

【選任した代理人】

    【識別番号】 100101306

    【弁理士】

    【氏名又は名称】 丸山 幸雄

    【電話番号】 03-5276-3241

【選任した代理人】

    【識別番号】 100115071

    【弁理士】

    【氏名又は名称】 大塚 康弘

【電話番号】 03-5276-3241

【手数料の表示】

【予納台帳番号】 003458

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0001010

【プールの可否】 要

【書類名】 明細書

【発明の名称】 ユーザインタフェース制御装置および方法ならびに記憶媒体

【特許請求の範囲】

【請求項 1】 ユーザインタフェースを介して入力される所定の制御対象に対する設定データの間を生じる不整合を回避するユーザインタフェース制御装置であって、

不整合回避記述を示すコンフリクト処理ルールを記憶する記憶手段と、

前記記憶手段に記憶された前記コンフリクト処理ルールに基づき、補完的な不整合回避記述を示す補完ルールを生成する補完ルール生成手段と、

前記コンフリクト処理ルールと前記補完ルールとに従い、入力された設定データを更新する更新手段と、

を有することを特徴とするユーザインタフェース制御装置。

【請求項 2】 前記補完ルール生成手段は、

前記コンフリクト処理ルール記述ファイルに、2 状態（ON および OFF）を有する前記制御対象の 1 の機能について、一方の状態に対するコンフリクト処理ルールが複数記述され、他方の状態に対するコンフリクト処理ルールは記述されていないとき、前記一方の状態に対するコンフリクト処理ルールの逆論理を、他方の状態に対するコンフリクト処理ルールに対する補完ルールとして生成することを特徴とする請求項 1 に記載のユーザインタフェース制御装置。

【請求項 3】 前記記憶手段は、

前記コンフリクト処理ルールを、コンフリクト処理ルール記述ファイルとして記憶することを特徴とする請求項 1 に記載のユーザインタフェース制御装置。

【請求項 4】 前記コンフリクト処理ルール記述ファイルは、所定のマークアップ言語に従って記述されることを特徴とする請求項 3 に記載のユーザインタフェース制御装置。

【請求項 5】 前記コンフリクト処理ルール記述ファイルには、特定の制御対象にのみ適用可能なローカルルールが記述され、複数の制御対象群に共通して適用可能な普遍ルールを記述したユニバーサルルール記述ファイルを外部参照とすることを特徴とする請求項 4 に記載のユーザインタフェース制御装置。

【請求項6】 前記コンフリクト処理ルール記述ファイルは、前記ユーザインタフェースの更新コマンドの記述を含むことを特徴とする請求項3ないし5のいずれか1項に記載のユーザインタフェース制御装置。

【請求項7】 前記補完ルール生成手段は、  
前記生成した補完ルールを前記コンフリクト処理ルール記述ファイルに追記する手段を更に有することを特徴とする請求項3ないし6のいずれか1項に記載のユーザインタフェース制御装置。

【請求項8】 前記更新手段により前記コンフリクト処理ルールまたは前記補完ルールの適用を受けて設定データが更新されたことを通知する手段を更に有することを特徴とする請求項1ないし7のいずれか1項に記載のユーザインタフェース制御装置。

【請求項9】 前記制御対象は、画像形成装置であることを特徴とする請求項1ないし8のいずれか1項に記載のユーザインタフェース制御装置。

【請求項10】 ユーザインタフェースを介して入力される所定の制御対象に対する設定データの間を生じる不整合を回避するユーザインタフェース制御方法であって、

不整合回避記述を示すコンフリクト処理ルールを記述したコンフリクト処理ルール記述ファイルを参照し、該コンフリクト処理ルールに基づいて、補完的な不整合回避記述を示す補完ルールを生成する補完ルール生成工程と、

前記コンフリクト処理ルールと前記補完ルールとに従い、入力された設定データを更新する更新工程と、

を有することを特徴とするユーザインタフェース制御方法。

【請求項11】 前記補完ルール生成工程は、  
前記コンフリクト処理ルール記述ファイルに、2状態（ONおよびOFF）を有する前記制御対象の1の機能について、一方の状態に対するコンフリクト処理ルールが複数記述され、他方の状態に対するコンフリクト処理ルールは記述されていないとき、前記一方の状態に対するコンフリクト処理ルールの逆論理を、他方の状態に対するコンフリクト処理ルールに対する補完ルールとして生成することを特徴とする請求項10に記載のユーザインタフェース制御方法。



【請求項 12】 前記コンフリクト処理ルール記述ファイルは、所定のマークアップ言語に従って記述されることを特徴とする請求項 10 に記載のユーザインタフェース制御方法。

【請求項 13】 前記コンフリクト処理ルール記述ファイルには、特定の制御対象にのみ適用可能なローカルルールが記述され、複数の制御対象群に共通して適用可能な普遍ルールを記述したユニバーサルルール記述ファイルを外部参照とすることを特徴とする請求項 12 に記載のユーザインタフェース制御方法。

【請求項 14】 前記コンフリクト処理ルール記述ファイルは、前記ユーザインタフェースの更新コマンドの記述を含むことを特徴とする請求項 10 ないし 13 のいずれか 1 項に記載のユーザインタフェース制御方法。

【請求項 15】 前記補完ルール生成工程は、  
前記生成した補完ルールを前記コンフリクト処理ルール記述ファイルに追記する工程を更に有することを特徴とする請求項 10 ないし 14 のいずれか 1 項に記載のユーザインタフェース制御方法。

【請求項 16】 前記更新工程により前記コンフリクト処理ルールまたは前記補完ルールの適用を受けて設定データが更新されたことを通知する工程を更に有することを特徴とする請求項 10 ないし 15 のいずれか 1 項に記載のユーザインタフェース制御方法。

【請求項 17】 ユーザインタフェースを介して入力される所定の制御対象に対する設定データの間を生じる不整合を回避するユーザインタフェース制御方法をコンピュータによって実現するためのプログラムを格納した記憶媒体であって、

不整合回避記述を示すコンフリクト処理ルールを記述したコンフリクト処理ルール記述ファイルと、

前記コンフリクト処理ルール記述ファイルを参照して、該コンフリクト処理ルールに基づき補完的な不整合回避記述を示す補完ルールを生成する補完ルール生成工程のプログラムコードと、

前記コンフリクト処理ルールと前記補完ルールとに従い、入力された設定データを更新する更新工程のプログラムコードと、

を格納することを特徴とする記憶媒体。

【請求項 1 8】 前記補完ルール生成工程のプログラムコードは、  
前記コンフリクト処理ルール記述ファイルに、2 状態（ONおよびOFF）を有する前記制御対象の 1 の機能について、一方の状態に対するコンフリクト処理ルールが複数記述され、他方の状態に対するコンフリクト処理ルールは記述されていないとき、前記一方の状態に対するコンフリクト処理ルールの逆論理を、他方の状態に対するコンフリクト処理ルールに対する補完ルールとして生成することを特徴とする請求項 1 7 に記載の記憶媒体。

【請求項 1 9】 前記コンフリクト処理ルール記述ファイルは、所定のマークアップ言語に従って記述されることを特徴とする請求項 1 7 に記載の記憶媒体。

【請求項 2 0】 前記コンフリクト処理ルール記述ファイルには、特定の制御対象にのみ適用可能なローカルルールが記述され、複数の制御対象群に共通して適用可能な普遍ルールを記述したユニバーサルルール記述ファイルを外部参照とすることを特徴とする請求項 1 9 に記載の記憶媒体。

【請求項 2 1】 前記コンフリクト処理ルール記述ファイルは、前記ユーザインタフェースの更新コマンドの記述を含むことを特徴とする請求項 1 7 ないし 2 0 のいずれか 1 項に記載の記憶媒体。

【請求項 2 2】 前記補完ルール生成工程のプログラムコードは、  
前記生成した補完ルールを前記コンフリクト処理ルール記述ファイルに追記する工程のプログラムコードを更に有することを特徴とする請求項 1 7 ないし 2 1 のいずれか 1 項に記載の記憶媒体。

【請求項 2 3】 前記更新工程により前記コンフリクト処理ルールまたは前記補完ルールの適用を受けて設定データが更新されたことを通知する工程のプログラムコードを更に有することを特徴とする請求項 1 7 ないし 2 2 のいずれか 1 項に記載の記憶媒体。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、ユーザインタフェースを介して入力される所定の制御対象に対する設定データの間に生じる不整合を回避するユーザインタフェース制御装置および方法ならびに記憶媒体に関する。

【0002】

【従来の技術】

ユーザインタフェース（以下、「UI」ともいう。）を介してユーザから複数の設定値の入力を受け付け、それらの設定値に基づき制御される装置の一例として、画像形成装置（プリンタ装置）がある。プリンタ装置は一般に、印刷動作を制御するプリンタドライバを備え、プリンタドライバは、ユーザからの印刷設定等を受け付けるUIを含む。

【0003】

プリンタドライバは、UIを介してユーザから設定値の入力を受け付ける度に、それまでに設定された複数の設定値の中で関連する設定値の値との関係进行评估し、設定値間の不整合（コンフリクト）がないかどうかを判別する。コンフリクトの例としては、記録媒体としてセットされたOHPシートに対して両面印刷を行わせるような設定といったユーザにとって不都合と予想される設定や、プリンタに不可能な動作を行わせるような設定等がある。

【0004】

コンフリクトがあった場合には、それを解消するためのコンフリクト処理を実施する必要がある。

【0005】

従来は、コンフリクトの判別およびコンフリクト処理を設定値間の関係に依存したかたちで行う専用のコンフリクト処理プログラムを用いるのが一般的であった。あるいは、コンフリクト処理が必要となる複数の設定値の条件を一覧としてまとめてファイル等に保存しておき、このファイルをコンフリクト処理プログラムに読み込ませることで、コンフリクト処理プログラムが特定の設定値に依存することを排除し、コンフリクト処理プログラムが汎用的に利用できるようになっている場合もある。

【0006】

## 【発明が解決しようとする課題】

しかしながら、いずれの場合もその実現のためには、プログラム開発者等がすべてのコンフリクト処理ルールを網羅的に記述する必要があった。そのため、設定値間の依存関係が複雑な場合には完全に条件を網羅することができずに、漏れが生じてしまう場合があるという問題がある。

## 【0007】

また、従来は、ルール記述のベースは組み合わせであり、1対1の対象機能制御しかできなかった。ルールを追加する場合には、入力者が記述全体をチェックする必要があった。そして、すべての組み合わせを網羅するようデータを作成する必要があるので入力量が非常に多い。また、ルールを1箇所に記載するため反復記述が多く、誤入力の可能性が高いうえ、多大な訂正工数がかかるという問題があった。

## 【0008】

本発明は上記の問題点に鑑みてなされたもので、漏れのない確実なコンフリクト処理を実現するとともに、コンフリクト処理ルールの記述方法を改善して、プログラム開発者等による入力工数や人為的ミスを低減することを可能とするユーザインタフェース制御装置および方法ならびに記憶媒体を提供することを目的とする。

## 【0009】

## 【課題を解決するための手段】

上記目的を達成するため、例えば本発明のユーザインタフェース制御装置は、以下の構成を備える。すなわち、

ユーザインタフェースを介して入力される所定の制御対象に対する設定データの間を生じる不整合を回避するユーザインタフェース制御装置であって、

不整合回避記述を示すコンフリクト処理ルールを記憶する記憶手段と、

前記記憶手段に記憶された前記コンフリクト処理ルールに基づき、補完的な不整合回避記述を示す補完ルールを生成する補完ルール生成手段と、

前記コンフリクト処理ルールと前記補完ルールとに従い、入力された設定データを更新する更新手段と、

を有することを特徴とする。

【0010】

【発明の実施の形態】

(ハードウェア構成)

図1は本発明の一実施形態を示す印刷処理システムのブロック構成図である。印刷処理システムは、ホストコンピュータ3000と、プリンタ1500より構成される。

【0011】

ホストコンピュータ3000において、1はシステムバス4に接続される各デバイスを総括的に制御するCPU、2はCPU1の主メモリ、ワークエリア等として機能するRAMである。3は各種プログラム、データを格納するROMであって、各種フォントを記憶するフォントROM3a、ブートプログラムやBIOS等を記憶するプログラムROM3a、および各種データを記憶するデータROM3cに区別して構成されている。

【0012】

5はキーボードコントローラ(KBC)で、キーボード9や不図示のポインティングデバイスからのキー入力を制御する。6はCRTコントローラ(CRTC)であり、CRTディスプレイ(CRT)10の表示を制御する。

【0013】

外部メモリ11(ディスクコントローラ(DKC)7によりアクセス制御される)は、ハードディスク(HD)やフロッピーディスク(FD)等であり、図示の如く、オペレーティングシステムプログラム(以下OS)205をはじめ各種アプリケーション(例えば、図形、イメージ、文字、表等が混在した文書処理を行う文書処理アプリケーションプログラム)201、印刷処理関連プログラム204を記憶する他、ユーザファイル、編集ファイル等も記憶する。印刷処理関連プログラム204は、プリンタ制御コマンド生成モジュール(以下「プリンタドライバ」という)2041、プリンタドライバUI制御モジュール2042を含む。

【0014】

8はプリンタコントローラ（PRTC）であり、双方向性インタフェース21を介してプリンタ1500に接続されて、プリンタ1500との通信制御処理を行う。

【0015】

外部メモリ11に記憶されたアプリケーションは、RAM2にロードされてCPU1により実行されることになる。また、CPU1は、例えばRAM2へのアウトラインフォントの展開（ラスタライズ）処理を実行し、CRT10上でのWYSIWYG（What you see is What you get）を可能としている。さらに、CPU1は、CRT10上の不図示のマウスカーソル等で指示されたコマンドに基づいて登録された種々のウインドウを開き、種々のデータ処理を実行する。ユーザは印刷を実行する際、印刷設定設定画面（プリンタドライバUI制御モジュール2042により制御される）を開き、プリンタの設定や、印刷モードの選択を含むプリンタドライバ2041に対する印刷処理の設定を行うことができる。

【0016】

プリンタ1500において、12はプリンタ1500の全体を制御するCPUである。19はCPU12の主メモリ、ワークエリア等として機能するとともに、出力情報展開領域、環境データ格納領域、NVRAM等に用いられるRAMであり、図示しない増設ポートに接続されるオプションRAMによりメモリ容量を拡張することができるように構成されている。13はROMであり、各種フォントを記憶するフォントROM13a、制御プログラム等を記憶するプログラムROM13b、および各種データを記憶するデータROM13cより構成される。

【0017】

外部メモリ14（メモリコントローラ（MC）20によりアクセスを制御される）は、オプションとして接続されるハードディスク（HD）、フロッピーディスク（FD）、ICカード等であり、フォントデータ、エミュレーションプログラム、フォームデータ等を記憶する。ハードディスク等の外部メモリ14が接続されていない場合には、ROM13のデータROM13cに、ホストコンピュータ3000で利用される情報等を記憶することになる。なお、外部メモリ14は1個に限らず、複数備えるものであってもよく、例えば、内蔵フォントに加えて

オプションフロントカード、言語系の異なるプリンタ制御言語を解釈するプログラム等を格納した外部メモリを複数接続できるように構成されていてもよい。

#### 【0018】

操作部1501にはユーザからの操作を受け付ける操作パネルが設けられ、その操作パネルには操作のためのスイッチおよびLED表示器等が配されている（図示は省略）。また、図示しないNVRAMを有し、操作パネル1501からのプリンタモード設定情報を記憶するようにしてもよい。

#### 【0019】

プリンタCPU12は、ROM13のプログラムROM13bに記憶された制御プログラム等に基づき、印刷部インタフェース16を介してシステムバス15に接続される印刷部（プリンタエンジン）17に出力情報としての画像信号を出力する。また、CPU12は入力部18を介してホストコンピュータ3000との通信処理が可能となっており、プリンタ1500内の情報等をホストコンピュータ3000に通知可能に構成されている。

#### 【0020】

##### （ソフトウェア構成）

図2は、所定のアプリケーションおよび印刷処理関連プログラムを起動して、ホストコンピュータ3000上のRAM2にロードされた状態のRAM2のメモリマップを示している。RAM2には、図示の如く、BIOS206、OS205をはじめ、アプリケーション201、印刷処理関連プログラム204、および関連データ203がロードされているとともに、空きメモリ領域202も確保されている。これにより、アプリケーション201および印刷処理関連プログラム204は実行可能状態にある。

#### 【0021】

印刷処理関連プログラム204におけるプリンタドライバUI制御モジュール2042は、ユーザによる印刷設定指令に応じてCRT10にプリンタドライバUIとしての印刷設定画面を表示しユーザからの設定を可能にする。

#### 【0022】

図8に、印刷設定画面の表示例を示す。同図において、[Print Style]欄80

は、印刷レイアウトを指定する欄であり、ユーザは例えば、片面印刷（1-Sided Printing）801、両面印刷（2-Sided Printing）802、および製本印刷（Booklet Printing）803のうちのいずれかを指定することができる。

【0023】

[Finishing] 欄 81 は、印刷済み記録媒体の出力順序および仕上げについて指定する欄であり、ユーザは例えば、以下のいずれかから指定可能となっている。

【0024】

[Collate] 811

部単位印刷。N ページの文書をM部印刷する場合に、第1 ページ、第2 ページ、…、第N ページの順に1枚ずつ出力し、これをM回繰り返す。

【0025】

[Group] 812

ページ単位印刷。N ページの文書をM部印刷する場合に、第1 ページをM枚、第2 ページをM枚、…、第N ページをM枚、の順に出力する。

【0026】

[Staple] 813

ステープル仕上げ。上記 [Collate] 811 と同様に部単位で出力し、仕上げとして各部ごとにステープラで止める。

【0027】

なお、本明細書では、上記したようなユーザ設定可能項目を「プリンタ機能」または単に「機能」とよぶ。この他にも多くのプリンタ機能を有するが、説明を簡単にするため省略する。

【0028】

ここで、ユーザにとって不都合と思われる設定の組み合わせ、意味のない設定の組み合わせ、すなわち設定値間の不整合（コンフリクト）は、プリンタドライバUI制御モジュール2042により、以下詳細に説明するコンフリクト処理として回避されるように設計される。例えば、図示においては、印刷レイアウトとして片面印刷（1-Sided Printing）801が指定されているが、この場合には、[Finishing] 欄 81 の [Staple] 813 は薄い灰色で表示され指定できないように



される。また、図9に示すように、印刷レイアウトとして製本印刷 (Booklet Printing) 803が指定されたときは、[Finishing]欄81のすべてが指定できないようにされる。上記した例はごく簡単な例示であって、実際に想定されるコンフリクトはかなりの数にのぼるであろう。以下、コンフリクト処理の詳細を説明する。

#### 【0029】

図3は、実施形態における印刷処理関連プログラム204のプリンタドライバUI制御モジュール2042の概略構成を示している。303は、各モジュール間のデータの受け渡しやデータの更新等を管理してコンフリクト処理を統括するコンフリクトマネージャである。306が、上記した印刷設定画面表示としてのプリンタドライバUIである。301は、後述する記述形式で記述される不整合回避記述を示すコンフリクト処理ルールを列記したコンフリクト処理ルール記述ファイルである。302は、コンフリクト処理ルール記述ファイル301をロードして新たなコンフリクト処理ルールを生成する推論エンジン、304は、各プリンタ機能の状態をリスト形式で表示する状態変数リストであり、ユーザからの入力およびコンフリクト処理ルール記述ファイル301の内容に基づき更新される。305は、プリンタドライバUI306が提供する画面表示の基になる帳票としての内部構造体であり、状態変数リスト304と連動して各プリンタ機能の状態を所定の形式で表示する。

#### 【0030】

コンフリクト処理ルール記述ファイル301には、骨格となる主たるルールが開発者によりあらかじめ列記される。推論エンジン302は、後ほど詳細に説明する方法で新たなコンフリクト処理ルールを自動生成し、コンフリクト処理ルール記述ファイル301に追記する。

#### 【0031】

その後、プリンタドライバUI306を介してユーザからの設定情報を受け取ったコンフリクトマネージャ303は、コンフリクト処理ルール記述ファイル301を参照する。このことは、図示の如くコンフリクト処理ルール記述ファイル301からコンフリクトマネージャ303に向かう矢印で、「R(Read)」とし

て表示されている。参照の結果、設定情報がコンフリクト処理ルールに適合する場合、コンフリクト処理が適用される。そうして、コンフリクトマネージャ303は、状態変数リスト304および内部構造体305を更新してプリンタドライバUI306に反映させる。この更新作業は、図示の如くコンフリクトマネージャ303は、状態変数リスト304および内部構造体305とは各々双方向矢印で結ばれ、「R/W (Read/Write)」として表示されている。

#### 【0032】

図4は、図3に示した各モジュールで扱われるデータの関連を説明する模式図である。図4において、コンフリクト処理ルール記述ファイル301は、推論エンジン302にインクルード（ロード）されたかたちで参照され、さらに新たなルールが追記される。このコンフリクト処理ルール記述ファイル301は、コンフリクトマネージャ303にも参照され、それを受けて状態変数リスト304が変更されることになる。また、内部構造体305と状態変数リスト304とは先に述べたとおり連動表示されるものであるから、互いにマッピングされる関係にある。そしてこの状態がプリンタドライバUI306によってユーザの目に見えるかたちで表現される。

#### 【0033】

内部構造体305は、プリンタ機能名A, B, C に対応するメンバをそれぞれ cA, cB, cC と表現する。

#### 【0034】

(コンフリクト処理ルールの記述形式)

次に、コンフリクト処理ルール記述ファイル301について説明する。

#### 【0035】

従来、機能名1対1の記述や機能名群を{}でくくる記述方法は、いずれも組み合わせをベースとしていた。そのため網羅的な記述が必要であった。先述したとおり、この問題を解決するため、主たるルールを開発者が記述することとし、そこから類推される例えば逆のルール等は推論エンジン302によって自動生成させる（詳細は後述）。

#### 【0036】

ルールの記述形式の概要は、次のとおりである。

【0037】

- ・ 宣言的知識をロジックで表現する。
- ・ 論理（ロジック）を用いてコンフリクト処理ルールを数学的に形式化する。
- ・ 知識は、普遍的な知識（例えば、複数の制御対象群に共通して適用可能な知識）とローカルな知識（例えば、特定の制御対象にのみ適用可能な知識）に分類できる。普遍的な知識はインクルード可能とする。
- ・ 論理のANDは記載する。ORは複数のルールに分割し排除する。NOTは使用可能とする。
- ・ 機能は引数を1つ持った述語の形で記述する。
- ・ 他の記述から導出できることについては重複記述しない。

【0038】

この概要から各ルールの記述方法を具体化する。各ルールの記述の基本的な形式は次のとおりである。

【0039】

- ・ 左辺には機能名(ON)、機能名(OFF)、機能名(値)を記入する。
- ・ 機能が成り立つ場合の論理を記入するときは、(ON)に対するすべてのルールを記述する。(OFF)に対するルールの記述は不要である（後述するように自動生成されるため）。
- ・ 機能が成り立たない場合の論理を記入するときは、(OFF)に対するすべてのルールを記述する。(ON)に対するルールの記述は不要である（自動生成されるため）。
- ・ 右辺には左辺が成り立つための論理を、機能名(ON)、機能名(OFF)、機能名(値)を記入する。複数の項を記述可能。また論理否定のNOTを使用可能である。

【0040】

上記したように、各ルールは、論理（ロジック）を用いてコンフリクト処理ルールを数学的に形式化する。述語は、「プリンタ機能名（引数）」の形で記述される。引数としては、ON/OFFの他、数値が入る場合がある（例えば、印刷部数等）。左辺にはプリンタ機能名（引数）を、右辺には左辺が成り立つための論理を

記述し、記号「←」または「<-」で関係づける。例えば、

$A(ON) \leftarrow B(ON).$

は、「プリンタ機能Bの状態がONのときは、プリンタ機能Aの状態をONとする」という意味のルールになる。

#### 【0041】

また、式中の記号「,」は「かつ (AND)」の意味で用いられる。例えば、「プリンタ機能Bの状態がON、かつ、プリンタ機能Cの状態がOFFのとき、プリンタ機能Aの状態をONとする」というルールは、

$A(ON) \leftarrow B(ON), C(OFF).$

と記述される。

#### 【0042】

上記した論理式の具体的な表記法は、宣言/論理型言語に準拠する形式で記述することもできるが、一部の表記方法や右左辺を反転しても同形式とみなすことができる。機能名、ON/OFFや()の表記の方法は、適宜設計し、またネットワーク経由で交換することも考慮し、マークアップ言語内に記述できるよう定義することもできる（マークアップ言語内での記述例については後述する）。

#### 【0043】

図6は、以上の例に従って記述されている。同図において、プリンタ機能として、図8に示した[Collate] 811に対応する部単位印刷機能、[Group] 812に対応するページ単位印刷機能、および[Staple] 813に対応するステープル仕上げ機能はそれぞれ、Collate()、Group()、およびStaple()で示され、引数はONまたはOFFとなる。また、[PrintStyle] 欄80に対応する印刷レイアウト機能は、Layout()で示され、引数は、1-Sided、2-Sided、Bookletのいずれかである。

#### 【0044】

図6の(1)は、ユーザにより[Group] 812をチェックされたことでGroup(ON)となったときは、Collate(OFF)とするルールを表す。(2)は、Staple(ON)となったときは、同じくCollate(OFF)とするルールである。また、(3)は、Layout(Booklet)となったときは、Group(OFF)とするルールを示している。

## 【0045】

(コンフリクト処理ルールの自動生成)

・開発者は、ある機能名に対して、ONとなるすべてのルールを記述し、上記したとおり、OFFとなるルールを省略することができる。OFFとなるルールは推論エンジン302により自動生成される。

・逆に、開発者は、ある機能名に対して、OFFとなるすべてのルールを記述し、上記したとおり、ONとなるルールを省略することができる。ONとなるルールは推論エンジン302により自動生成される。

・開発者は、ある機能名に対して、ONとなるルール、OFFとなるルールのすべてを記述することもできる。この場合ルールは自動生成されない。

・左辺に機能名(ON)、機能名(OFF)の形式で記述された項に対して、右辺に項目を記述する場合には、左辺と同一のON/OFF形式にする。この場合にルールが自動生成される。

## 【0046】

以下、代表的な表記方法を用いて説明する。

## 【0047】

ここで、論理式における左辺と右辺の必要十分条件的関係について補足説明しておく。

## 【0048】

$$A(ON) \leftarrow B(ON). \quad (a)$$

というルールが1行だけ記述されていた場合には、 $B(ON)$ は $A(ON)$ に対して十分条件であり、逆に $A(ON)$ は $B(ON)$ に対して必要条件となる。したがって、下記のような逆の条件のルールは一般的には逆はまた真なりとはならない。

$$B(ON) \leftarrow A(ON). \quad (b)$$

$$A(OFF) \leftarrow B(OFF). \quad (c)$$

## 【0049】

上記(a)、(b)、(c)が同時に成り立つ場合には、 $A(ON)$ と $B(ON)$ は互いに必要十分条件の関係にある。(a)に対して論理的な対偶である下記のルール

$$B(OFF) \leftarrow A(OFF). \quad (d)$$

は必ず真である。したがって、上記(a)、(b)、(c)が同時に成り立つ場合には、(a)か(d)のいずれか一方が開発者により記載されていれば、推論エンジン 3 0 2 により自動的に正しいコンフリクト条件（ルール）が生成されることになる。

## 【 0 0 5 0 】

次に、処理ルールの記述方法と自動生成されるロジックの関係について説明する。

## 【 0 0 5 1 】

2 状態値(ON,OFF)を値として持つルールで、同じ機能名について複数行記述されている場合には、原則として左辺にONまたはOFFの一方のみを記述する。例えば、

## 【 0 0 5 2 】

A(ON) ← B(ON),C(OFF).

A(ON) ← D(V1).

B(OFF) ← E(OFF).

ただし、引数V1は、数値を表す。この場合のルールに対して推論エンジン 3 0 2 は、次に示すON/OFF逆側のルールを補完ルールとして自動生成する。

## 【 0 0 5 3 】

A(OFF) ← true. (e)

B(ON) ← true.

## 【 0 0 5 4 】

この補完ルールは下記のルールを最適化したものである。

## 【 0 0 5 5 】

A(OFF) ← not A(ON).

B(ON) ← not B(OFF).

## 【 0 0 5 6 】

これは論理としてはA(ON)とA(OFF)は完全に排他な関係にあることを意味している。つまりA(ON)とA(OFF)でAについての集合空間が100%埋め尽くされる。B(ON)とB(OFF)も同様である。結果、A(ON/OFF)が失敗することはなくなり、ON/OFFのいずれかが成立する。

## 【0057】

ユーザがA(ON)とA(OFF)を混在して記述する場合には(e)の自動生成はされない。このような場合にはAの集合空間を埋め尽くすように記述する必要がある。

## 【0058】

処理ルールには機能名( )以外にプライオリティやアクションを記述できる組み込み述語を用意する。以下、組み込み述語と関連記述方法について説明する。

## 【0059】

(プライオリティおよびアクションの記述)

プライオリティの記述に用いる組み込み述語の代表例を挙げておく。

## 【0060】

status(機能名、値)

引数として指定した機能名の現在値が指定した値のときはtrue、そうでないときはfalseを返す。

## 【0061】

また、ルールに対してそのアクションを記述することができる。代表例を挙げておく。

## 【0062】

右辺に、{ }で囲み、ルールが成り立ったときに実行されるアクションを記述する。{ }内の組み込み述語としてはメッセージを表示するMessage()や、コントロールを制御するEnable,Disable,Show,Hide等が使用可能である。プライオリティやアクションは、オプショナルの表現形式としてはルールの形式に合わせて設計すればよい。

## 【0063】

(組み込み述語とデフォルト値)

組み込み述語status(A,X)によって機能Aの状態変数値を変数\_Xに受けることができる。推論エンジン302は、コンフリクト処理ルール記述ファイル301をロードした後、出現するすべてのルール名について次のルールを自動的に生成する。

## 【0064】

$A\_X \leftarrow \text{status}(A, X).$

$B\_X \leftarrow \text{status}(B, X).$

$C\_X \leftarrow \text{status}(C, X).$

...

$A(\text{ON}) \leftarrow B(\text{ON}), C(\text{OFF})$

【0065】

Aの状態値は上記ルールを受けてONになる。Bがもし上記自動生成された以外のルールが存在しないならば、

$B\_X \leftarrow \text{status}(B, X).$

が適用される。この自動生成ルールは必ず成り立つので、Bの状態変数の値ONが\_XにユニファイされルールBの状態値となる。

【0066】

(状態変数の充足機構)

確認した状態変数に関連する全ルールに対して充足処理を行う。

【0067】

例)

$C(\text{ON}) \leftarrow A(\text{ON}).$

$B(\text{OFF}) \leftarrow A(\text{ON}).$

$A(\text{ON}).$

Aの状態変数を確認した場合、AにONが充足され、それを参照しているB,CにそれぞれOFF,ONが充足される。確認した状態変数に関連する全ルールに対して充足処理をする。

【0068】

(拘束理由の設定)

sreason(R)の使用によって、状態変数の結果がそうなった理由を設定することができる。

【0069】

例)

$B(\text{OFF}) \leftarrow A(\text{OFF}), \{ \text{sreason}(R) \}.$



AがOFFであるとBもOFFになる場合の理由をRに設定しておく。コンフリクト等が発生した場合の理由を後に取り出すことができる。例えば、コンフリクト処理ルール記述ファイル301に、

A(ON) ← B(ON), C(OFF).

が記述されている場合、図4の状態変数リスト304に示すように、コンフリクト処理ルール記述ファイル301の中に出現するプリンタ機能A, B, Cのそれぞれについて同名の状態変数が存在している。

#### 【0070】

(プリンタドライバUI制御モジュール2042の処理の内容)

以下、図5のフローチャートを用いて、コンフリクト処理を含むプリンタドライバUI制御モジュール2042の処理について、詳しく説明する。

#### 【0071】

プリンタドライバUI制御モジュール2042の処理は、ユーザがキーボードコントローラKBC5等を用いて、プリンタドライバUIを開く指示をすることで始まる。ユーザがプリンタドライバUIを開くよう指示すると、上記したとおり、OS205の管理の下、RAM2に印刷処理関連プログラム204がロードされる。ここで印刷処理関連プログラム204は、ページ記述言語を用いて記述される印刷データを生成するプログラムであるため、同系列の複数のプリンタに対して共通に利用されるモジュールである。そのため、印刷要求時には、本印刷処理関連プログラム204は、ユーザがプリンタドライバUIを開くよう指示したプリンタドライバUIを起動する必要がある。

#### 【0072】

印刷処理関連プログラム204がRAM205にロードされると、まず、プリンタドライバUIを開くための初期化处理として、推論エンジン302は、コンフリクト処理ルール記述ファイル301をコンフリクトマネージャ303を介して、RAM2に読み込む(ステップS501)。

#### 【0073】

続いて、ステップ501で読み込んだコンフリクト処理ルールを元に、2状態値(ONとOFF)を値として持つルールについて、補完ルールを新たに生成する(

ステップS502)。ここで例を示す。コンフリクト処理ルール記述ファイル301が、以下のように記述されていたとする。

【0074】

$A(ON) \leftarrow B(ON), C(OFF). \quad \dots (f)$

$A(ON) \leftarrow D(V1). \quad \dots (g)$

$B(OFF) \leftarrow E(OFF). \quad \dots (h)$

【0075】

この場合、(f)と(g)はともに、同一のプリンタ機能Aについて、同一の状態ONについての記述がされており、他の式、ここでは(h)、においては、A(OFF)についての記述はない。このように、左辺に、同じプリンタ機能について、ONまたはOFFのどちらかだけが、コンフリクト処理ルール記述ファイル301に記述されている場合、上述したとおり、推論エンジン302は、当該式に対してON/OFF逆側のルールを自動生成する。したがって、この例の場合、(f)、(g)に対しては

$A(OFF) \leftarrow \text{not } A(ON). \quad \dots (i)$

が生成され、(h)に対しては、

$B(ON) \leftarrow \text{not } B(OFF). \quad \dots (j)$

が生成される。この自動生成された(i)、(j)のルールは処理効率の観点から最適化されて以下の(i)', (j)'のように変形される。ただし、意味は全く同じである。

$A(OFF) \leftarrow \text{true}. \quad \dots (i)'$

$B(ON) \leftarrow \text{true}. \quad \dots (j)'$

【0076】

上記した例のように逆側が自動生成された場合は、論理として考えた場合に、A(ON)とA(OFF)とは完全に排他的な関係にあることを意味している。つまりA(ON)とA(OFF)とでプリンタ機能Aについての集合空間が100%埋め尽くされることを意味している。B(ON)とB(OFF)も同様である。

【0077】

なお、A(ON)とA(OFF)が混在して記述されている場合には、(i)および(i)'

は自動生成されない。したがってこの場合には、すべてのケースを網羅的に記述してAについての集合空間を100%埋め尽くすようにする必要がある。

#### 【0078】

続いて、状態変数リスト304とコンフリクト処理ルール記述ファイル301で使用されるプリンタ機能名の値の初期化に関するコンフリクト処理ルールを、補完ルールとして自動生成する（ステップS503）。

#### 【0079】

コンフリクト処理ルール記述ファイル301に記述されるすべてのプリンタ機能名は、コンフリクトマネージャ303の内部にインクルードされた状態変数リスト304にそれぞれ、状態変数を持っている。この状態変数の値は、プリンタドライバUI306で使用される内部構造体305の対応するメンバの値と連動している。各プリンタ機能名の状態変数の初期値は、その内部構造体305のメンバの値となる。

#### 【0080】

例えば、図4において、内部構造体305に記述されているint cAの初期値は0であるから、それに対応する状態変数リスト304におけるプリンタ機能Aの値はOFFとなっている。したがって、推論エンジン302内に記述される補完ルールの、プリンタ機能Aの状態の初期値はOFFとなる。

#### 【0081】

その後、推論エンジン302は、コンフリクト処理ルール記述ファイル301を参照することでコンフリクトチェックの推論を行う。例えば、図4に示すように、コンフリクト処理ルール記述ファイル301に記述されている、

A(ON) ← B(ON), C(OFF).

が成立した場合、推論エンジン302は、状態変数リスト304の、プリンタ機能Aの状態変数値を、初期値OFFからONに変更する。コンフリクトチェックの推論が終了した後、コンフリクトマネージャは、変更された状態変数の値を内部構造体305の対応するメンバ int cAに反映させる。すなわち、int cA は上記ルールが成立したことによって0から1に変更される。

#### 【0082】

推論エンジン 302 は、組込み関数 `status(a, _X)` によってプリンタ機能 A の状態変数値を推論エンジン 302 で使用される変数 `_X` に受けることができる。推論エンジン 302 はコンフリクト処理ルール記述ファイル 301 をロードした後、そのコンフリクト処理ルール中に出現する全てのルール名について、次のルールを自動的に生成する。

【0083】

`A(_X) ← status(A, _X).`

`B(_X) ← status(B, _X).`

`C(_X) ← status(C, _X).`

:

【0084】

これは、他に適用するルールが存在しない場合には、内部構造体 305 の対応するメンバの値がそのプリンタ機能名の状態値となることを意味する。

【0085】

A については、

`A(ON) ← B(ON), C(OFF).`

が成り立つので、A の状態値はこれを受けて ON となる。B がもし上記自動生成されたルール以外にルールが存在しないならば、

`B(_X) ← status(B, _X).`

が適用される。この自動生成ルールは必ず成り立つので、B の状態変数の値 ON が `_X` にユニファイされ、それがプリンタ機能 B の状態値となる。つまり、ユーザ定義ルールが存在しないか、または存在していても成り立つものがないプリンタ機能については、内部構造体 305 の対応するメンバに格納されている値がそのプリンタ機能の状態値ということになる。

【0086】

図 7 は、ステップ S502 およびステップ S503 によって生成されたルールを、図 6 に示した元のコンフリクト処理ルール記述ファイル 301 に追記した例を示している。ステップ S502 では、(1) および (2) から生成される (4) と、(3) から生成される (5) とが補完ルールとして追記される。さらに、ステップ S50

3では、(6)から(9)までが追記されることになる。

【0087】

続いて、プリンタドライバUI 306のオープンのために必要とされるその他の初期化処理を行い、図8に例示したようなプリンタドライバUIをオープンする（ステップS504）。

【0088】

プリンタドライバUI 306がオープンされた後は、OSより送られてくるイベントを取得し、そのイベントに対する処理を繰り返す（ステップS505）。

【0089】

次に、ステップS505にて取得したイベントが、ユーザがプリンタドライバUI 306上の設定項目を変更したイベントであるかどうかの判別を行い（ステップ506）、そうでなかった場合には、ステップS512に進み、プリンタドライバUI 306のクローズ要求かどうかの判別を行う。クローズ要求であった場合には、ステップS513に進み、終了処理を行い、プリンタドライバUI 306をクローズして、すべての処理を終了する。一方、ステップS512で、クローズ要求ではなかった場合は、ステップS505に戻り、処理を繰り返す。

【0090】

ステップS506で、ステップS505で取得したイベントがユーザによる設定変更要求であった場合には、ステップS507に進み、ステップS501からステップS503の処理により新たに生成したコンフリクト処理ルールを適用する。

【0091】

取得したイベントがユーザによる設定変更要求であった場合の一例として、図8に示す[Print Style]欄80における、片面印刷（1-Sided Printing）801から製本印刷（Booklet Printing）803に変更するものであった場合について説明する。このとき、内部構造体305のメンバとして存在するCollate、Group、Staple、Layoutの各メンバのコンフリクト処理ルール適用前、すなわち、設定変更要求前における値は、以下のようになっていた。

【0092】

Collate OFF

Group ON

Staple OFF

Layout 1-Sided

【0093】

ユーザの変更要求がLayoutを1-SidedからBookletに変更するものであるので、Layoutのメンバの内容が変更されて、内部構造体305の各メンバの値は次のようになる。

【0094】

Collate OFF

Group ON

Staple OFF

Layout Booklet

【0095】

すると、プリンタドライバUI306はコンフリクトマネージャ303をコールし、状態変数リスト304にあるLayoutの状態変数が更新され、続いて推論エンジン302がコールされて、コンフリクト処理ルールの適用が始まる。まず図7の(6)～(9)のルールが適用され、推論エンジン302内の各プリンタ機能名が状態変数リストの各メンバの持つ値で初期化される。続いて、図7の(3)が適用され、その結果、次のようにGroupの値はONからOFFへと変化する。

【0096】

Collate OFF

Group OFF

Staple OFF

Layout Booklet

【0097】

さらに、図7の(4)のルールが適用され、CollateがOFFからONへと変化する。

【0098】

Collate ON

Group OFF

Staple OFF

Layout Booklet

【0099】

この他に、適用されるルールが存在しなくなると、推論エンジン302でのコンフリクト処理ルールの適用が終了する。

【0100】

コンフリクトマネージャ303は、上記ステップS507のコンフリクト処理ルールをの適用結果に基づき状態変数リストを更新し（ステップS508）、内部構造体305を更新する（ステップS509）。

【0101】

続いて、プリンタドライバUI306が内部構造体305のメンバの値を参照して、UIの更新が必要かどうかの判別を行う（ステップS510）。UIの更新の必要がない場合には、ステップS505に戻り、処理を繰り返す。UIの更新が必要な場合には、UIの更新を行ったうえで（ステップS511）、ステップS505に戻り、処理を繰り返すことになる。上記の例では、Layoutが1-Sided PrintingからBooklet Printingに設定が変更されたことにより、CollateがOFFからONへ、GroupがONからOFFへと変化しているので、プリンタドライバUIは、図8から図9に示すとおり更新される。

【0102】

以上の処理は、プリンタドライバUI306がクローズされるまで、繰り返し実行される。プリンタドライバUI306がクローズされると処理は終了し、印刷処理関連プログラム204の処理も終了し、RAM2からはOS205の機能により消去される。

【0103】

ところで、プリンタドライバUIの更新処理の実行は、コンフリクト処理ルール記述ファイル301の中に、プリンタドライバUIを更新するための処理を記述し、推論エンジン302がその記述を解釈した時点で、コンフリクトマネージャ303の状態変数リスト304を介して、プリンタドライバUIの更新処理を

直接行うようにしても良い。

#### 【0104】

図10は、図7に示した追記後のコンフリクト処理ルールのうち、(3)の次の行に、UI更新の処理として {disable} の記述を追加したものである。この記述により、図9におけるGroupラジオボタンコントロールをdisableする処理（設定不可の状態にする処理）がコンフリクト処理ルールの適用内容の一部として実現されることになる。

#### 【0105】

さらに、図11に示すように、コンフリクト処理ルールの中に、ユーザに対する情報表示を可能にするメッセージボックスの表示処理を加えることも可能である。例えば、図11の {disable} の次の行の記述 { Message(MSG001) } は、図12に示すようなメッセージボックスを表示させることを示している。「MSG001」は、図12に表示されているメッセージテキスト「Groupの設定はCollateに調整されました。」の文字列を指示するIDで、ID: MSG001およびそのIDが示す文字列は、コンフリクトマネージャ303内に文字列リソースとして存在している。

#### 【0106】

次に、コンフリクト処理ルールの記述をマークアップ言語（例えば、XML (Extensible Markup Language) ）内におく具体例について説明する。

#### 【0107】

図13は、マークアップ言語内におけるコンフリクト処理ルールの記述例を示している。図示のように、コンフリクト処理ルール部は<コンフリクトルールズ>タグの間に記述され、各ルールは<ルール>タグで囲まれる。構造が指示されれば、タグ名は任意でかまわない。

#### 【0108】

また、コンフリクト処理ルールは、多くの機種種のプリンタに共通して適用可能なルール群（普遍（ユニバーサル）ルール）と、特定の機種種のプリンタにのみ適用可能なルール群（ローカルルール）とに分類することができよう。この場合、例えば、図示の如く、普遍ルールは<コンフリクトルールズ.ユニバーサル>タグで囲み、ローカルルールは<コンフリクトルールズ.ローカル>タグで囲むこ



とも可能である。

【0109】

さらに、図14に示すように、普遍ルールだけを記述したコンフリクトルールズ・ユニバーサルファイルを外部参照用ファイルとして作成し、コンフリクト処理ルール記述ファイルにインクルードするかたちにすることも可能である。

【0110】

以上説明したように、実施形態によれば、プログラム開発者などが用意するコンフリクト処理ルールを補完するコンフリクト処理ルールを自動生成するので、より品質の高いコンフリクト処理を実現することができる。

【0111】

また、ルール記述の基本をロジックでおこなうことにより、1対多の機能制御が可能となる。依存関係はロジックにより伝播するため、新しいルールの追加が容易である。ルール追加時も記述全体をチェックする必要はない。そして、ロジックの自動生成が行われるのでコンフリクトすべての組み合わせを網羅するようにデータを作成する必要はなくなる。また、汎用的なルールは別に用意できるため反復記述が抑えられ、誤入力と訂正工数を減らすことができる。

【0112】

さらに、ユーザインタフェースの更新処理、メッセージ処理もコンフリクト処理ルールに加えることで、開発者にとって可読性の高い、メンテナンスの容易なコーディングを実現することができる。

【0113】

また、コンフリクト処理ルールに、ユーザインタフェース更新処理とメッセージ処理を、コンフリクト処理とともに設けることで、コンフリクト処理ルールが変更された場合でも、ユーザインタフェース制御モジュール自体を変更する必要がなくなる。

【0114】

なお、上述した実施形態では、プリンタ装置に対して、コンフリクト処理を含むUI制御を行う例について説明したが、プリンタ装置に限らず、デジタルカメラ、デジタルレコーダ、イメージスキャナ等の周辺装置、制御機器の他、モ

デムやルータといったネットワーク関連機器にも適用可能であることはいうまでもない。

【0115】

【他の実施形態】

なお、本発明は、複数の機器（例えばホストコンピュータ、インタフェース機器、リーダ、プリンタなど）から構成されるシステムに適用しても、一つの機器からなる装置（例えば、複写機、ファクシミリ装置など）に適用してもよい。

【0116】

上述したように、本発明の目的は、前述した実施形態の機能を実現するソフトウェアのプログラムコードを記録した記憶媒体（または記録媒体）を、システムあるいは装置に供給し、そのシステムあるいは装置のコンピュータ（またはCPUやMPU）が記憶媒体に格納されたプログラムコードを読み出し実行することによっても、達成される。この場合、記憶媒体から読み出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。また、コンピュータが読み出したプログラムコードを実行することにより、前述した実施形態の機能が実現されるだけでなく、そのプログラムコードの指示に基づき、コンピュータ上で稼働しているオペレーティングシステム（OS）などが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【0117】

さらに、記憶媒体から読み出されたプログラムコードが、コンピュータに挿入された機能拡張カードやコンピュータに接続された機能拡張ユニットに備わるメモリに書込まれた後、そのプログラムコードの指示に基づき、その機能拡張カードや機能拡張ユニットに備わるCPUなどが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【0118】

本発明を上記記憶媒体に適用する場合、その記憶媒体には、先に説明した図5

に示すフローチャートに対応するプログラムコードが格納されることになる。

【0119】

【発明の効果】

以上説明したように、本発明によれば、漏れのない確実なコンフリクト処理を実現するとともに、コンフリクト処理ルールの記述方法を改善して、プログラム開発者等による入力工数や人為的ミスを低減することを可能とするユーザインタフェース制御装置および方法ならびに記憶媒体を提供することができる。

【図面の簡単な説明】

【図1】

実施形態に係る印刷処理システムのブロック構成図である。

【図2】

実施形態におけるRAM2のメモリマップを示す図である。

【図3】

実施形態におけるプリンタドライバUI制御モジュールの概略構成図である。

【図4】

実施形態におけるプリンタドライバUI制御モジュールで扱われるデータの関連を説明する模式図である。

【図5】

実施形態におけるプリンタドライバUI制御モジュールの処理を示すフローチャートである。

【図6】

実施形態におけるコンフリクト処理ルールの一例を示す図である。

【図7】

実施形態におけるコンフリクト処理ルールの一例を示す図である。

【図8】

実施形態における印刷設定画面の一例を示す図である。

【図9】

実施形態における印刷設定画面の一例を示す図である。

【図10】

実施形態におけるコンフリクト処理ルールの一例を示す図である。

【図 1 1】

実施形態におけるコンフリクト処理ルールの一例を示す図である。

【図 1 2】

実施形態におけるメッセージボックスの一例を示す図である。

【図 1 3】

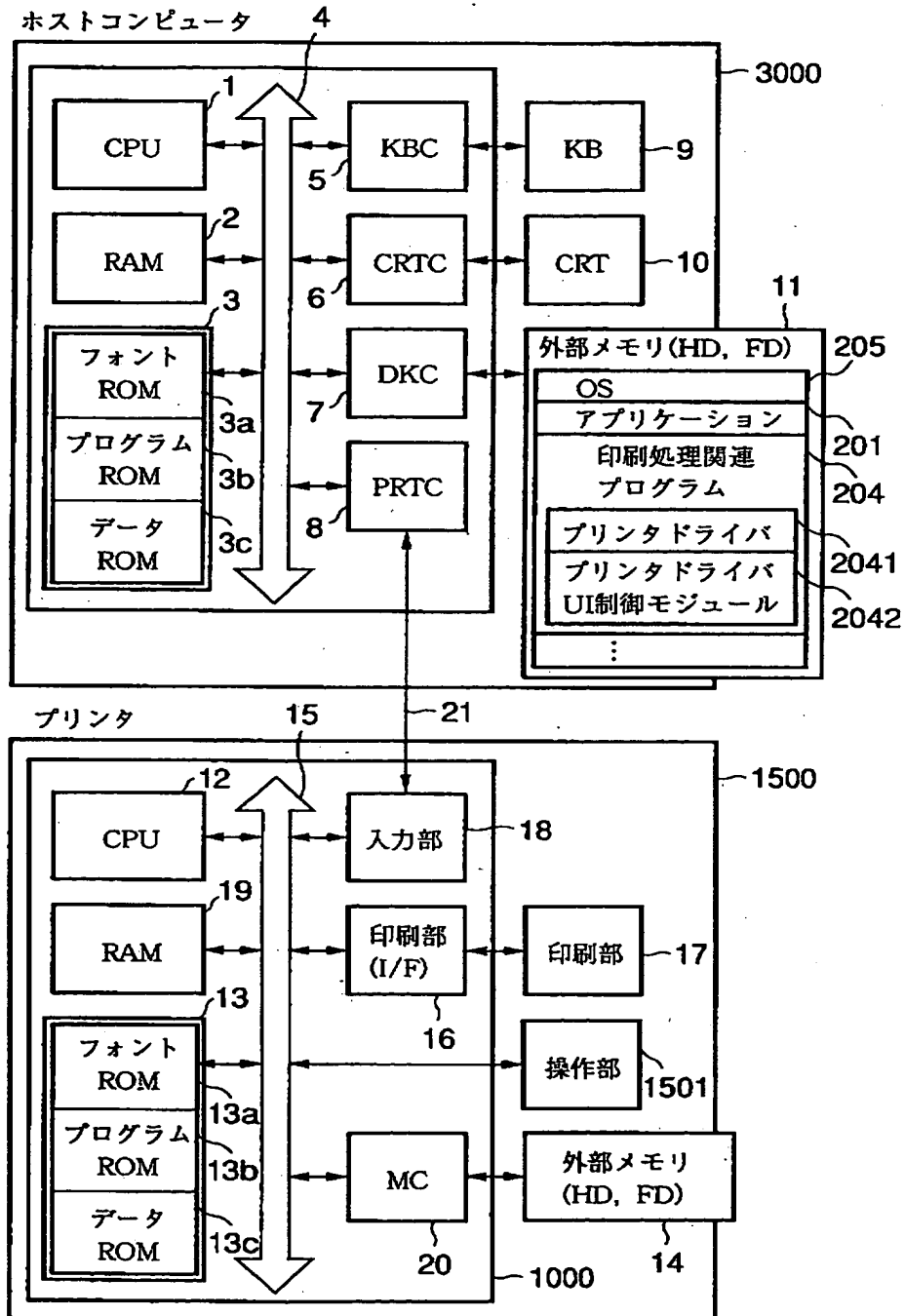
実施形態におけるマークアップ言語内におけるコンフリクト処理ルールの記述例を示す図である。

【図 1 4】

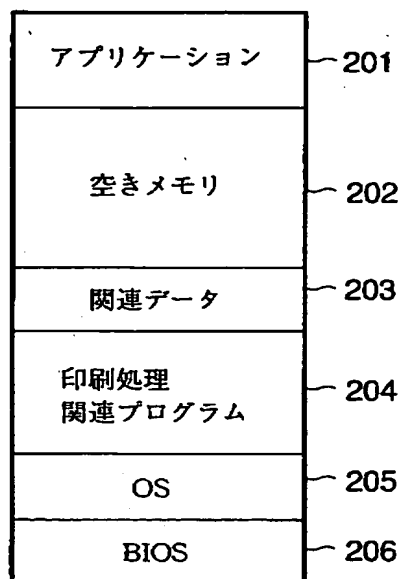
実施形態におけるマークアップ言語内におけるコンフリクト処理ルールの記述例を示す図である。

【書類名】 図面

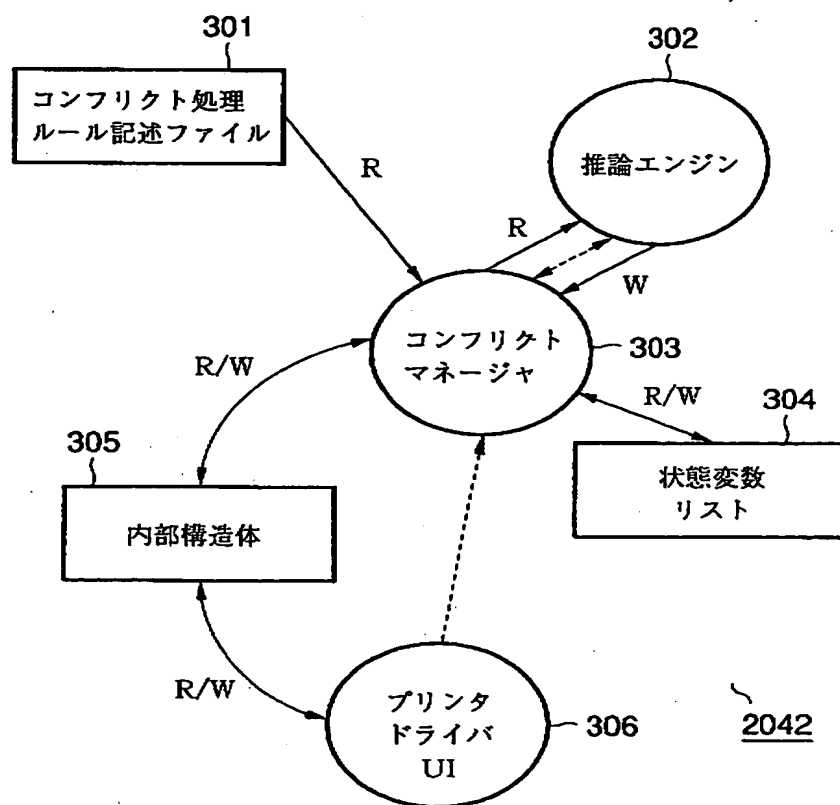
【図 1】



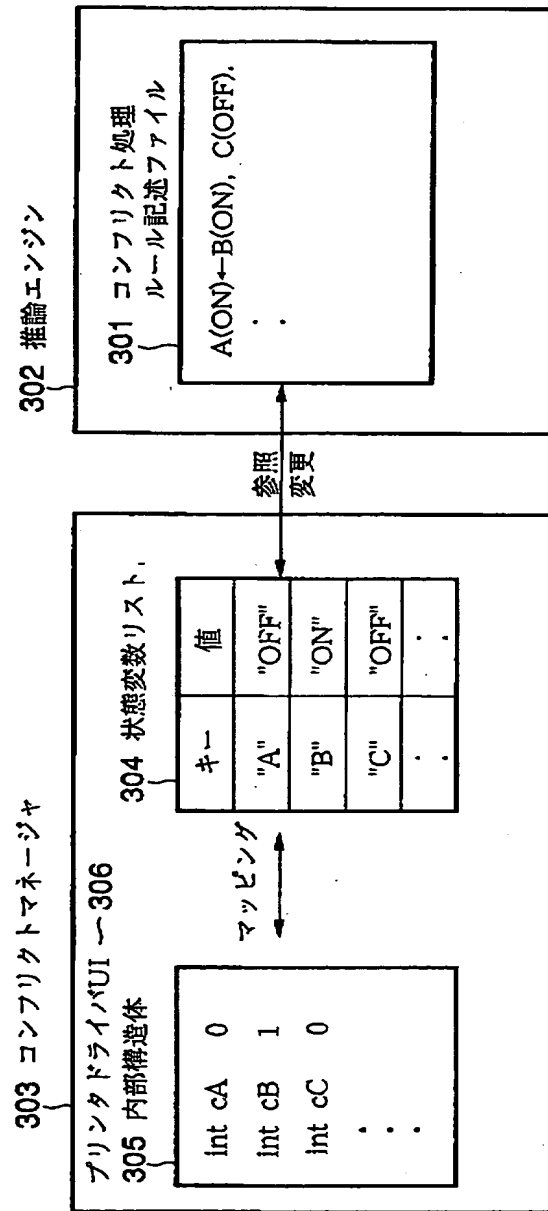
【図 2】



【図3】

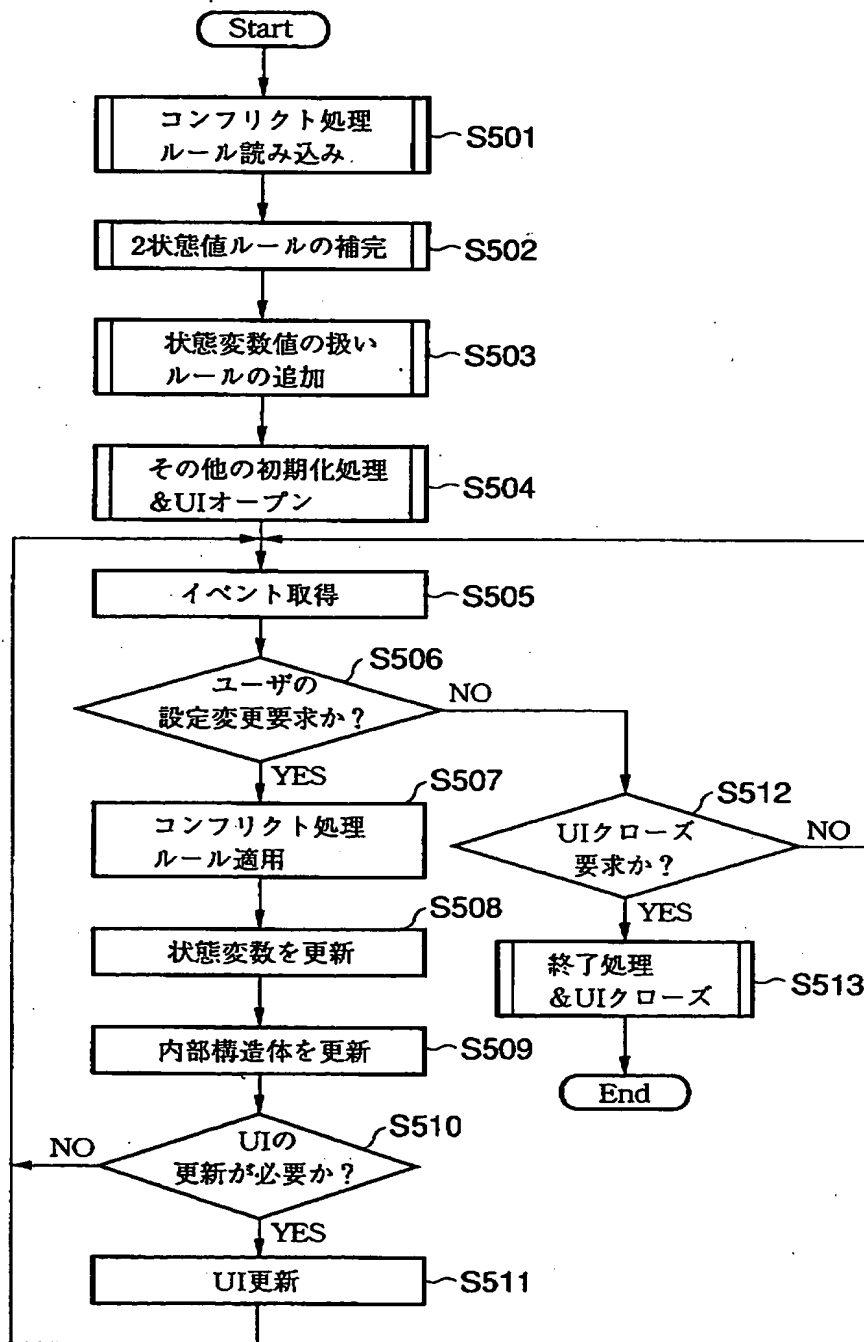


【図 4】





【図 5】



【図 6】

301

Collate(OFF) <- Group(ON).	(1)
Collate(OFF) <- Staple(ON).	(2)
Group(OFF) <- Layout(BOOKLET).	(3)

【図 7】

301

Collate(OFF) <- Group(ON).	(1)
Collate(OFF) <- Staple(ON).	(2)
Group(OFF) <- Layout(BOOKLET).	(3)
Collate(ON) <- true.	(4)
Group(ON) <- true.	(5)
Collate(_X) <- status(Collate,_X).	(6)
Group(_X) <- status(Group,_X).	(7)
Layout(_X) <- status(Layout,_X).	(8)
Staple(_X) <- status(Staple,_X).	(9)

【図8】

印刷設定画面

Print Style: ☒ 1-Sided Printing ~ 801  
☐ 2-Sided Printing ~ 802  
☐ Booklet Printing ~ 803 ~ 80

Finishing: ☐ Collate ~ 811  
☒ [Group] ~ 812 ~ 81  
☐ Staple ~ 813


Staple Position

OK キャンセル

【図9】


印刷設定画面

Print Style:



☐ 1-Sided Printing ~ 801  
☐ 2-Sided Printing ~ 802  
☒ [Booklet Printing] ~ 803

Finishing:



☒ Collate  
☐ Group  
☐ Staple

81

Staple Pillion

OK

キャンセル

【図 1 0】

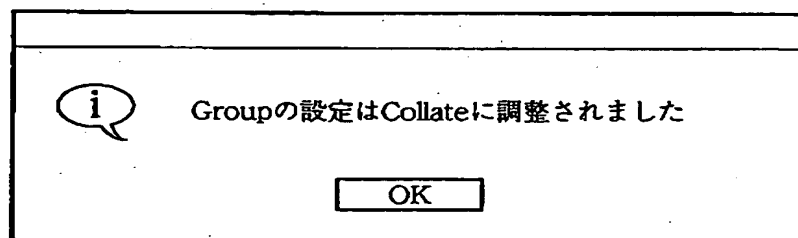
```
Collate(OFF) <- Group(ON).           (1)
Collate(OFF) <- Staple(ON).          (2)
Group(OFF) <- Layout(BOOKLET).        (3)
                                   {disable}
Collate(ON) <- true.                  (4)
Group(ON) <- true.                    (5)
Collate(_X) <- status(Collate,_X).    (6)
Group(_X) <- status(Group,_X).        (7)
Layout(_X) <- status(Layout,_X).      (8)
Staple(_X) <- status(Staple,_X).      (9)
```

【図 1 1】

```
Collate(OFF) <- Group(ON).           (1)
Collate(OFF) <- Staple(ON).          (2)
Group(OFF) <- Layout(BOOKLET).       (3)
    {disable}
    {Message(MSG001)}

Collate(ON) <- true.                  (4)
Group(ON) <- true.                    (5)
Collate(_X) <- status(Collate,_X).    (6)
Group(_X) <- status(Group,_X).        (7)
Layout(_X) <- status(Layout,_X).      (8)
Staple(_X) <- status(Staple,_X).      (9)
```

【図 12】





【図 13】

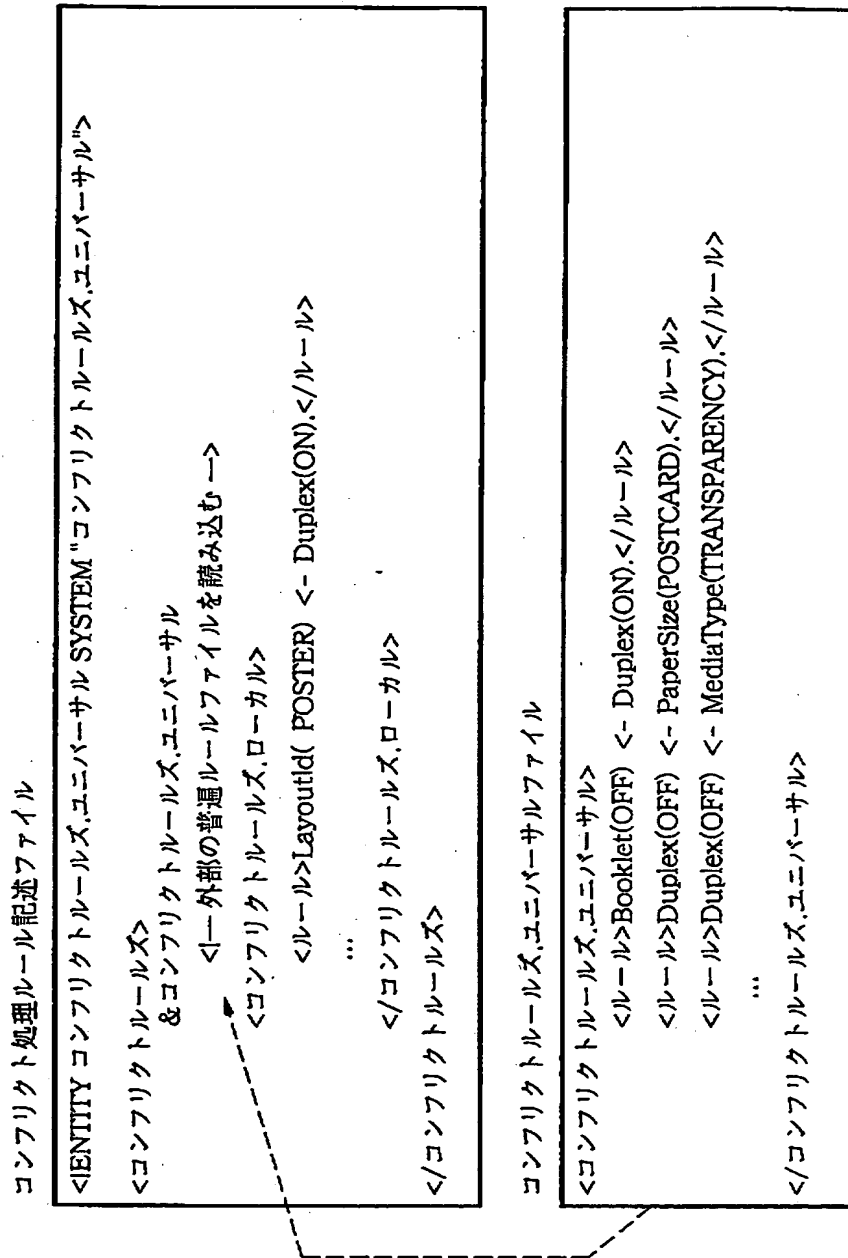
コンフリクト処理ルール

```

<コンフリクトルールズ>
  <コンフリクトルールズ,ユニバーサル>
    <ルール>Booklet(OFF) <- Duplex(ON).</ルール>
    <ルール>Duplex(OFF) <- PaperSize(POSTCARD).</ルール>
    <ルール>Duplex(OFF) <- MediaType(TRANSPARENCY).</ルール>
    ...
  </コンフリクトルールズ,ユニバーサル>
  <コンフリクトルールズ,ローカル>
    <ルール>LayoutId(~POSTER) <- Duplex(ON).</ルール>
    ...
  </コンフリクトルールズ,ローカル>
</コンフリクトルールズ>

```

【図 14】



【書類名】 要約書

【要約】

【課題】 漏れのない確実なコンフリクト処理を実現するとともに、コンフリクト処理ルールの記述方法を改善して、プログラム開発者等による入力工数や人為的ミスを低減することを可能とするユーザインタフェース制御装置および方法ならびに記憶媒体を提供すること。

【解決手段】 ユーザはあらかじめ、設定データの中に生じる不整合を回避するための記述を示すコンフリクト処理ルールを、コンフリクト処理ルール記述ファイル（301）に記述しておく。推論エンジン（302）は、このコンフリクト処理ルール記述ファイル（301）を参照し、補完的な不整合回避記述を示す補完ルールを生成する。補完ルールは、類推可能に記述されている機能について、当該記述の逆論理ルールを生成することにより実現される。

【選択図】 図3

出 願 人 履 歴 情 報

識別番号 [000001007]

1. 変更年月日 1990年 8月30日  
[変更理由] 新規登録  
住 所 東京都大田区下丸子3丁目30番2号  
氏 名 キヤノン株式会社